Reporte Técnico

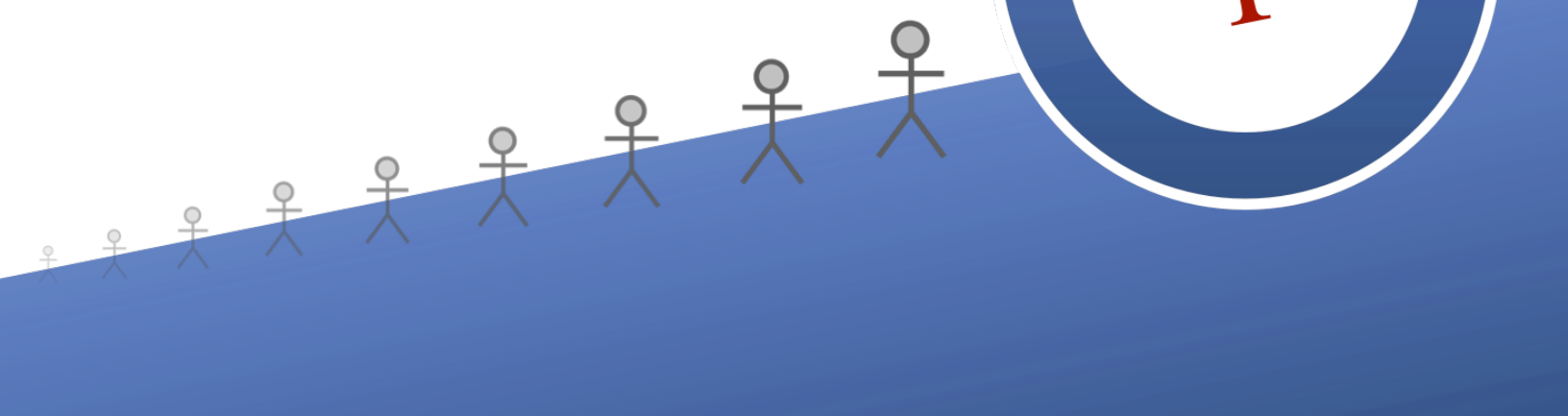
Prototipo de sistema de identificación de estudiantes basado en reconocimiento facial dentro de la Escuela Superior de Cómputo para evitar la suplantación de identidad durante la aplicación de Exámenes a Título de Suficiencia mediante el uso de credenciales y dispositivos móviles

Trabajo terminal A-138

De la Cruz De la Cruz Alejandra
Flores Esquivel Luis Antonio
Huertas Ramírez Daniel Martin
Jiménez Rodríguez José Alfredo

Escuela Superior de Cómputo, IPN

24 de noviembre de 2024



Índice general

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

Índice de figuras

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

Índice de cuadros

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

CAPÍTULO 1

Problemática

Presentar la problematica

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

CAPÍTULO 2

Marco Teórico

Este capítulo está enfocado en detallar la información esencial para el entendimiento del Trabajo Terminal, además de explicar y establecer las tecnologías que se usarán para el desarrollo de este. Para comenzar

2.1. Evaluaciones a Título de Suficiencia (ETS)

En el Instituto Politécnico Nacional (IPN), incluyendo unidades académicas como la Escuela Superior de Cómputo (ESCOM), la acreditación de cada unidad de aprendizaje se realiza semestralmente a través de 3 evaluaciones ordinarias. Si un alumno no acredita alguna unidad de aprendizaje, tendrá la oportunidad de presentar una evaluación extraordinaria. Estos procedimientos están detallados en el programa de estudios y se encuentran especificados en el calendario académico [**?**].

El alumno que no logre acreditar una o más de las unidades de aprendizaje en la que se haya inscrito podrá optar por acreditarlas mediante Evaluación a Título de Suficiencia, de acuerdo con lo establecido en el Artículo 39 del Reglamento Interno del IPN, que señala:

> "La evaluación del aprendizaje se llevará a cabo a través de exámenes ordinarios, extraordinarios y a título de suficiencia, cuyos requisitos y procedimientos de elaboración, presentación y exención, así como de otros mecanismos de evaluación continua, se realizarán en los términos que fijen los planes y programas de estudio, el presente Reglamento y los reglamentos respectivos."

Existen dos rondas de ETS:

- ETS Ordinario: Esta es la primera oportunidad que tiene el alumno para acreditar la materia en la que no obtuvo una calificación aprobatoria. Los ETS ordinarios generalmente se aplican al finalizar el semestre, permitiendo al estudiante demostrar sus conocimientos sin necesidad de repetir el curso completo.
- ETS Especiales: Si el alumno no acredita la materia en el ETS ordinario, puede optar por presentar un ETS Especial. Esta es la segunda oportunidad que tiene el alumno para pasar la materia. Esta evaluación adicional suele programarse el primer viernes del nuevo semestre, brindando al estudiante una opción rápida para regularizar sus situación académica y continuar avanzando en su plan de estudios.

2.1.1. Procedimiento para realizar un ETS

- Pagar en caja, verificar que estén correctos los siguientes datos: Nombre, Boleta, Carrera y Número de unidades de aprendizaje.
- Acudir a ventanilla de gestión escolar para generar créditos en el “SAES”.
- Una vez generados los créditos, inscribe las unidades de aprendizaje en la página del “SAES”.
- Entregar en ventanilla de gestión escolar, el comprobante de inscripción de ES generador por SAES, y el recibo de pago para dar fin a la inscripción al ETS.
- Acudir el día y la hora establecida en el calendario [**?**].

2.2. Códigos QR

Un código QR es un tipo de código de barras bidimensionales que solo se puede leer con teléfonos inteligentes u otros dispositivos dedicados a la lectura de estos códigos. Cuando se lee un código QR, los dispositivos se conectan directamente a mensajes de texto, correos electrónicos, sitios web, números de teléfono, etc [**?**].

2.2.1. Anatomía de un código QR

Patrones de detección de posición

Figura 2.1: Patrones de detección de posición.

Los patrones de detección de posición se encuentran en las tres esquinas del código. Gracias a ellos, el escáner puede reconocer y leer el código QR rápidamente. Estos marcadores indican la dirección en la que se imprimió el código QR y ayudan a su identificación y orientación.

Patrones de alineación
Usados para corregir la distorsión del código QR en superficies curvas. El tamaño y la cantidad de los patrones de alineación pueden variar según el volumen de la información almacenada en el código.

Patrones de temporización
La alternancia de los módulos negros y blancos del código QR detrmina el sistema de información, también llamado cuadrícula de datos. Con estas líneas, el escáner reconoce la matriz de datos.

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

Figura 2.2: Patrones de alineación.

Figura 2.3: Patrones de temporización.

Información sobre la versión

Figura 2.4: Información sobre la versión.

Estos marcadores indican cuál de las 40 versiones del código QR está siendo usada. Normalmente las versiones utilizadas son de 1 a 7.

Información del formato

Figura 2.5: Información del formato.

Contiene información sobre la tolerancia a los errores y el patrón del enmascaramiento de datos. La información sobre el formato facilita el escaneo del código.

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

Código de corrección de datos y errores

Figura 2.6: Código de corrección de datos y errores.

El sistema de corrección de errores del código QR almacena toda la información y comparte el espacio con los módulos de corrección de errores, que permiten reconstruir los datos perdidos.

Márgenes

Figura 2.7: Márgenes.

Los márgenes, o también llamado zona quieta, alrededor del código QR son similares al espacio blanco en un diseño, proporcionan estructura y una mejor comprensión. Pero, ¿cómo? Para que el software de escaneo identifique bien el límite del código QR de sus alrededores, los márgenes son vitales.

2.2.2. Fiabilidad de los códigos QR

Los códigos QR están diseñados para mantener la información legible, aunque estén oscuros o dañados. Esto se logra mediante la compensación de errores, es decir, insertando la información varias veces. Con un alto nivel de seguridad, los códigos QR pueden leerse incluso si un tercio de su información es ilegible. Esto hace que los códigos QR sean muy fiables a la hora de guardar información.

Para la elaboración de nuestro TT, el código QR desempeña un papel fundamental en la corroboración de la identidad de los estudiantes. Las credenciales escolares al contener un código QR se puede aprovechar la capacidad para

almacenar y transmitir información de manera segura y rápida. Este enfoque nos permite verificar la información del estudiante mediante un simple escaneo, reduciendo el tiempo necesario para corroborar la identidad. Al escanear el código QR en la credencial. El sistema accede a los datos del alumno, lo cual permite verificar si coincide con la persona que se presenta al ETS.

2.3. Aplicación móvil

Una aplicación móvil, es un tipo de aplicación diseñada para ejecutarse en un dispositivo móvil, que puede ser un teléfono inteligente. A diferencia de las aplicaciones diseñadas para computadoras de escritorio, las aplicaciones móviles se alejan de los sistemas de software integrados.

Debido a los recursos de hardware limitados de los primeros dispositivos móviles, las aplicaciones móviles evitaban la multifuncionalidad. Sin embargo, incluso si los dispositivos que se utilizan hoy en día son mucho más sofisticados, las aplicaciones móviles siguen siendo funcionales. Así es como los propietarios de aplicaciones móviles permiten a los consumidores seleccionar exactamente las funciones que deben tener sus dispositivos.

Existen diferentes tipos de aplicaciones móviles que responden a las necesidades y preferencias de los usuarios, así como a las capacidades técnicas de los dispositivos. A continuación, se detallan cada uno:

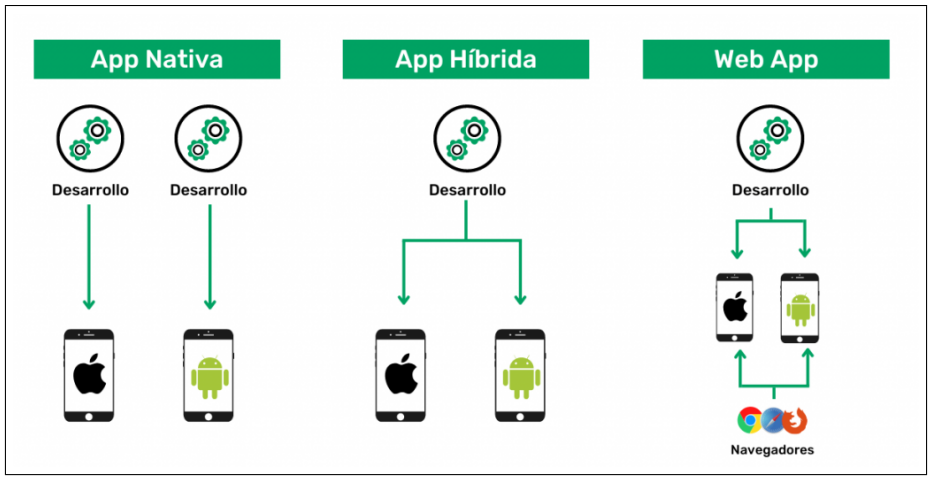


Figura 2.8: Tipo de aplicaciones moviles.

2.3.1. Aplicaciones nativas

Las aplicaciones nativas son apps desarrolladas para un sistema operativo móvil concreto (¡OS o Android normalmente), en el lenguaje de programación específico de cada plataforma. Esto quiere decir que una app nativa creada para Android no puede ser utilizada en un dispositivo IOS y viceversa.

Es el tipo de aplicación móvil más conocida. Para que funcione, debemos descargarla desde los markets de apps, como App Store o Google Play e instalarla en nuestro teléfono [**?**].

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

Ventajas

- **Tienen el mejor rendimiento.** Las aplicaciones nativas son las más rápidas y tienen un rendimiento superior a otros tipos de apps, ya que han sido optimizadas específicamente para el hardware y el sistema operativo del dispositivo.
- **Acceso completo e integración con las funciones hardware del dispositivo.** Las apps nativas permiten aprovechar al máximo las funcionalidades móviles: cámara, micrófono, lector biométrico de huella, sensores y redes inalámbricas.
- **Pueden funcionar sin acceso a internet (funcionamiento offline)** si han sido diseñadas para ello.

Desventajas

- **Costes de desarrollo altos.** Si queremos tener nuestra app disponible para los dos sistemas, necesitaremos dos líneas de desarrollo diferentes, ya que el código utilizado para un sistema no es reutilizable para otro.
- **Complejidad de desarrollo.** Necesitamos equipos expertos en el lenguaje específico de cada sistema. Por ejemplo, en Kotlin para Android y en Swift para IOS.
- **Tiempo de desarrollo superior.** El desarrollo puede tomar entre 4 a 6 meses.

2.3.2. Aplicaciones Web

Las aplicaciones web realmente son webs especiales diseñadas para navegadores móviles. A diferencia de las apps nativas o híbridas, no necesitan ser descargadas, ya que se accede a ellas desde un navegador web.

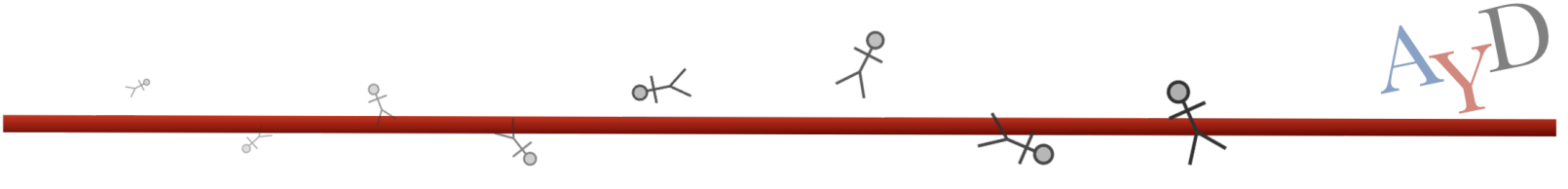
Emplean las mismas tecnologías de desarrollo que una web, como HTML, CSS o JavaScript. Así, estaríamos hablando de una web con apariencia de app, por lo que presentaría sus mismas limitaciones. Sin embargo, con la llegada del HTML5, se han conseguido salvar algunas limitaciones, como el acceso a algunas funciones del móvil (geolocalización, cámaras).

Ventajas

- **Carácter multiplataforma.** Con una sola línea de desarrollo.
- **Fácil desarrollo.** Se emplean tecnologías ampliamente conocidas.
- **Tiempo y coste de desarrollo bajo.**

Desventajas

- **Acceso limitado a las funciones del dispositivo.**
- **No se pueden subir a las tiendas de aplicaciones.**
- **Diferentes experiencias de usuario.** Estas dependen del navegador utilizado.
- **Necesidad de conexión a Internet.** Incluso si se cuenta con un modo pensado para ello, es necesario para acceder a las posibles actualizaciones o para entrar por primera vez.



2.3.3. Aplicaciones Híbridas

Las aplicaciones híbridas o multiplataforma combinan elementos de las aplicaciones nativas y las aplicaciones web. Estas aplicaciones se desarrollan utilizando tecnologías web como HTML, CSS y JavaScript, pero se empaquetan en un formato que puede ser instalado en un dispositivo móvil como cualquier otra aplicación nativa. Por tanto, podemos obtener una aplicación para varias plataformas con un único desarrollo.

Ventajas

- **Menor coste.** Gracias al uso de lenguajes de programación más conocidos, con una mayor disponibilidad de profesionales en el mercado.
- **Carácter multiplataforma.** Con una sola línea de desarrollo.
- **Acceso a algunas funcionalidades del móvil.**
- **Reducción de los tiempos de desarrollo.** Generalmente, el tiempo de desarrollo se reduce a 3 meses.
- **Disponibilidad en markets.** Se pueden subir a los markets de aplicaciones, como App Store y Google Play.

Desventajas

- **Rendimiento inferior.** Su rendimiento es inferior al de una app nativa, suelen tener un tamaño considerable y, además, ser más lentas.
- **Acceso limitado a las funciones del dispositivo.**

2.3.4. Aplicaciones Progresivas Web Apps (PWA)

Las aplicaciones progresivas son un reciente avance de las Web Apps. Al igual que las Web Apps, son webs diseñadas para móviles, pero esta vez, sí pueden ser descargadas en el móvil como una aplicación más, aunque no es necesario para que ofrezcan un comportamiento similar al de una app nativa a través del navegador.

Las PWA adoptan un comportamiento más propio de aplicaciones nativas que de web, como el funcionamiento sin Internet, un mayor rendimiento o su funcionamiento en segundo plano. Sin embargo, como desventaja, seguimos contando con la imposibilidad de subirlas a los markets de aplicaciones.

Para comprender mejor las diferencias entre los tipos de aplicaciones móviles, a continuación se presenta una tabla comparativa que destaca sus características clave:

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

Tipos de app	Nativa	Híbrida	Web
Interfaz	Basada en web	Específica de la plataforma (iOS, Android)	Basada en web
Tiempo de desarrollo	Alto	Medio	Bajo
Coste de desarrollo	Alto	Medio	Bajo
Multiplataforma	No	Sí	Sí
Rendimiento	Alto	Medio	Bajo
Acceso a los sensores del dispositivo	Completo	Alto o Completo	Limitado
Tiendas de aplicaciones	Sí	Sí	No

Cuadro 2.1: Comparación tipos de aplicaciones móviles.

Para el desarrollo de nuestro trabajo terminal, hemos decidido optar por una aplicación híbrida con el uso de Kotlin. Esta decisión se basa en varios factores relacionados con los recursos disponibles, las características de nuestro público objetivo y los plazos establecidos.

La aplicación móvil está dirigida para estudiantes, alumnos y personal de seguridad de la Escuela Superior de Cómputo, donde la mayoría utiliza dispositivos con sistema operativo Android. La elección de una aplicación híbrida nos permite optimizar la experiencia en Android, que es la plataforma que predomina entre nuestros usuarios.

Aunque las aplicaciones híbridas suelen desarrollarse con tecnologías web (como React Native o Flutter), para nuestro proyecto hemos decidido incorporar Kotlin para desarrollar modelos donde se requiera un rendimiento nativo o un acceso más profundo a las funciones del sistema operativo Android.Además, las aplicaciones híbridas permiten un desarrollo más rápido en comparación con las aplicaciones completamente nativas, ya que gran parte del código puede compartirse entre plataformas, así mismo, es una buena opción económica que se adapta a nuestro presupuesto.

2.4. Sistema operativo

2.5. Lenguaje de programación

2.6. Framework

2.6.1. Jetpack compose

Para la implementación de la aplicación móvil que forma parte de nuestro sistema de identificación y control de acceso, hemos decidido utilizar Jetpack Compose. La elección de la tecnología adecuada es importante para ofrecer una mejor experiencia de usuario y cumplir nuestros objetivos de diseño y funcional.

¿Por que Jetpack compose?

Jetpack compose es un framework (estructura o marco de trabajo que, bajo parámetros estandarizados, ejecutan tareas específicas en el desarrollo de un software) con la particularidad de ejecutar prácticas modernas en los desarrolladores de software a partir de la reutilización de componentes, así como también contando con la oportunidad de crear animaciones y temas oscuros. En este sentido, Jetpack Compose es el conjunto de herramientas ofrecidas por Android para el desarrollo de aplicaciones con un objetivo específico: simplificar y optimizar los códigos en la IU nativas [**?**].

Ventajas

- **Menos código:** Simplifica el proceso de desarrollo haciendo menos código, todo se basa en funciones de modo que el código será simple y fácil de mantener.
- **Intuitiva:** Tan solo describe tu IU con un enfoque declarativo haciendo “qué hay que hacer” en vez de “cómo se debe hacer”.
- **Potente:** Tiene integrado Material Design con el cual puede crear apps atractivas al usuario con animaciones y mucho más.
- **Acelera el desarrollo:** Es compatible con proyectos existentes, puedes empezar a integrarlo por partes cuando quieras y donde quieras.
- **Kotlin:** Está escrito 100 % en Kotlin, lo cual nos permitirá usar sus herramientas potentes y API’s intuitivas.

2.6.2. XML

Para este proyecto utilizaremos XML debido a la múltiples ventajas que ofrece en el desarrollo de interfaces de usuario en aplicaciones web y Android.

¿Por que XML?

XML son siglas de *Extensible Markup Language*, es un lenguaje de marcado que proporciona reglas para definir cualquier dato.

Por ejemplo, imaginemos un documento de texto con comentarios. Los comentarios pueden ofrecer sugerencias como las siguientes:

- Ponga el título en negrita.
- Esta oración es un encabezado.
- Esta palabra es del autor.

Estos comentarios mejoran la usabilidad del documento sin repercutir en su contenido. Del mismo modo, XML utiliza símbolos de marcado para proporcionar más información sobre los datos.

Etiquetas XML

Los símbolos de marcado, denominados **etiquetas** en XML, se utilizan para definir los datos. Por ejemplo, para representar los datos de una librería, se pueden crear etiquetas como:

`<libro>`, `<título>` y `<autor>`

El documento XML de un solo libro tendría el siguiente contenido:

```
<libro>
<titulo>Introduccion a Amazon Web Services</titulo>
<autor>Mark Wilkins</autor>
</libro>
```

Las etiquetas ofrecen una sofisticada codificación de datos para integrar los flujos de información en diferentes sistemas.

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

Ventajas de XML

- **Flexibilidad:** El formato XML es un lenguaje de marcas que se puede personalizar para diferentes propósitos.
- **Interoperabilidad:** El formato XML es compatible con una amplia gama de sistemas y aplicaciones, lo que significa que los datos se pueden intercambiar fácilmente entre diferentes sistemas.
- **Legibilidad:** El formato XML es fácil de leer y entender, lo que facilita la creación y el mantenimiento de archivos XML.
- **Reutilización:** Los elementos y atributos de un archivo XML se pueden reutilizar en diferentes partes del archivo, lo que ahorra tiempo y reduce errores.

2.7. Patrones de arquitectura de software

Al momento de desarrollar software, es común toparnos con problemáticas que requieren de la toma de decisiones especialmente cuando hablamos sobre cuestiones relacionadas con el diseño de un sistema de software. Un patrón de arquitectura de software es un conjunto de decisiones tomadas para atacar problemáticas relacionadas con el diseño de un software. Estos incluyen reglas y principios para organizar las interacciones entre subsistemas predefinidos y los roles que estos desempeñan [1].

A menudo pueden ser descritos como los *"Planos"* de un sistema, sin embargo, esto no quiere decir que sea la arquitectura final, sino que funcionan como una guía que describe los elementos necesarios para diseñar la arquitectura de la solución a desarrollar, la selección de una arquitectura sobre otra dependerá completamente de los objetivos a alcanzar, los recursos disponibles y la experiencia del equipo de desarrollo involucrado [2].

Es necesario aclarar que, los patrones de arquitectura no deben confundirse con los patrones de diseño, ya que ambos responden a problemas diferentes durante el desarrollo de un sistema de software, de forma muy breve un patrón de arquitectura describe como crear la lógica de negocio, acceso a los datos, etc. Mientras que los patrones de diseño se usan al implementar estos elementos [1].

A continuación, se describen algunos patrones de arquitectura que pueden ser de utilidad al implementar la propuesta de solución para este trabajo terminal.

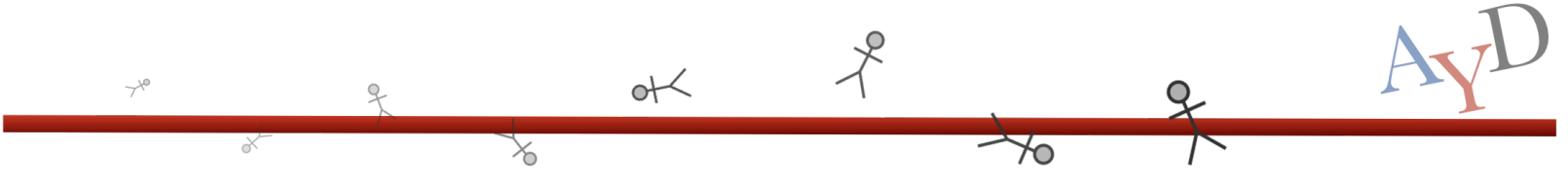
2.7.1. Arquitectura de capas

También conocida como *arquitectura de N-capas*, estructura una aplicación en múltiples capas distintas, donde cada una esta encargada de ciertas tareas en especifico, lo que permite dividir un sistema en componentes aislados, lo que facilita el desarrollo rápido de aplicaciones ya que los cambios realizados en una capa no deberían afectar la lógica de las demás [3].

Las arquitecturas basadas en este patrón suelen implementar cuatro capas distintas, la capa de presentación, la capa de negocio, de persistencia y de base de datos, sin embargo, y como es de esperarse, la arquitectura de un sistema basado en este patrón puede tener algunas diferencias en el número y tipo de capas que se implementan, por ejemplo, algunas pueden implementar capas de aplicación, servicio o acceso a datos [1].

2.7.2. Arquitectura orientada a servicios

Las aplicaciones diseñadas siguiendo este patrón de arquitectura implementan una colección de servicios poco acoplados que se comunican entre sí a través de una red. Cada uno de los servicios que conforman al sistema se encarga de llevar acabo una función del negocio en específico que después pueden ser requeridos por otro servicio o cliente [3].



2.7.3. Arquitectura de micro servicios

Es una arquitectura que combina patrones de diseño para crear múltiples servicios que trabajan de forma independiente y que en conjunto forman la lógica de una aplicación. Es una alternativa a las aplicaciones monolíticas y de la arquitectura basada en servicios, en donde cada servicio esta orientado a implementar partes muy puntuales de la lógica de negocio [2].

Su principal ventaja radica en que debido a que sus componentes se encuentran poco acoplados pueden ser desarrollados, desplegados y probados de forma independiente [3].

La principal desventaja de este tipo de arquitectura es que puede llegar a ser compleja de implementar ya que requiere de definir la granularidad correcta de los servicios y establecer una comunicación efectiva entre estos [1].

2.7.4. El patrón modelo vista controlador

Este patrón divide divide una aplicación en tres componentes interconectados. El modelo, la vista y el controlador. Esta separación permite organizar el código al desacoplar la lógica del negocio, interfaz de usuario y el manejo de las entradas de los usuarios, lo que a su vez promueve la modularidad, mantenibilidad y escalabilidad [3].

El modelo contiene los datos y lógica de negocio de la aplicación. Se encarga de regresar, almacenar y procesar la información.

La vista, también conocida como interfaz de usuario (UI) despliega la información al usuario y responde a las interacciones del usuario.

El controlador funciona como un intermediario entre el modelo y la vista. Se encarga de gestionar las entradas del usuario, actualizar el modelo y la vista para reflejar los cambios realizados en el modelo [2].

2.8. Redes neuronales artificiales

Una red neuronal artificial es una técnica para la creación de programas de computación que son capaces de aprender de los datos. Se basa en el conocimiento actual sobre como funciona el cerebro humano. Consisten en un conjunto de nodos o neuronas interconectados entre sí que procesan y aprenden de los datos con los que cuentan, lo que permite llevar a cabo tareas como el reconocimiento de patrones y la toma de decisiones en el aprendizaje de máquina [1].

2.8.1. Estructura de una red neuronal artificial

Toda red neuronal esta formada por capas de nodos, o neuronas artificiales, una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo se conecta a los demás y tiene un peso y un umbral determinado. A continuación se describe el papel que cada una de estas capas en la arquitectura de una red neuronal común [2].

Capa de entrada

La capa de entrada es responsable de recibir los datos iniciales en forma de vectores. Cada neurona en esta capa corresponde a una característica del conjunto de datos [3].

Capas ocultas

Las capas ocultas son el núcleo de las RNA y donde ocurre el procesamiento de los datos. Estas capas aplican transformaciones no lineales mediante funciones de activación, como *ReLU*, *sigmoide*, o *tangente hiperbólica*. El número y tamaño de las capas ocultas determinan la capacidad de la red para modelar patrones complejos [3].

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

Capa de salida

La capa de salida genera los resultados finales del modelo. Para tareas de clasificación, por ejemplo, esta capa utiliza funciones como *softmax* para proporcionar probabilidades asociadas a cada clase [3].

2.9. Proceso de aprendizaje

Como ya se mencionó con anterioridad, una red neuronal es un sistema complejo que busca imitar la forma en la que el cerebro humano aprende [1]. Este proceso de aprendizaje se realiza en dos fases conocidas como *backpropagation* y *forward propagation* [3]:

2.9.1. Forward propagation

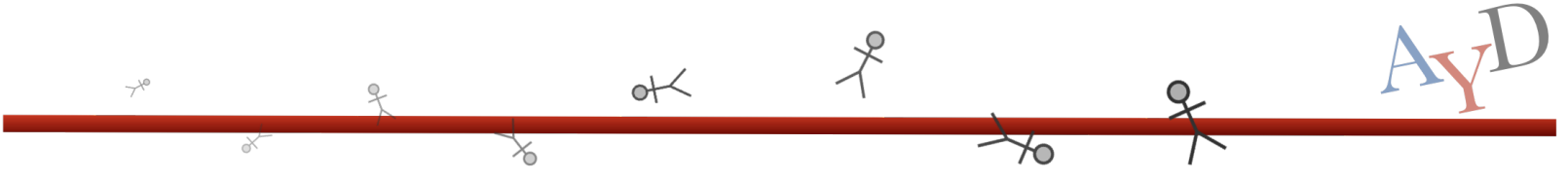
- **Capa de entrada:** La capa de entrada contiene nodos que representan cada característica del conjunto de datos inicial. Estos nodos reciben y procesan los datos de entrada.
- **Pesos y conexiones:** Los pesos asignados a cada conexión entre neuronas determinan la influencia de una neurona sobre otra. Durante el entrenamiento, estos valores se actualizan constantemente para optimizar el rendimiento de la red.
- **Capas ocultas:** Las neuronas en las capas ocultas combinan las entradas recibidas al multiplicarlas por sus respectivos pesos y sumarlas. Posteriormente, aplican una función de activación que introduce no linealidad, lo que permite identificar relaciones complejas en los datos.
- **Capa de salida:** Este proceso se repite hasta llegar a la capa de salida, donde se genera el resultado final de la red neuronal.

2.9.2. Backpropagation

- **Cálculo del error:** La salida generada por la red se compara con los valores esperados utilizando una función de pérdida. Para problemas de regresión, se emplea comúnmente el *Error Cuadrático Medio* (*Mean Squared Error, MSE*), que calcula la diferencia entre las predicciones y los valores reales: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **Optimización por descenso de gradiente:** Para reducir el error, la red emplea el algoritmo de descenso de gradiente. Este ajusta los pesos calculando la derivada de la función de pérdida con respecto a cada peso, guiando los ajustes necesarios para minimizar el error.
- **Ajuste de pesos:** Este proceso de actualización de pesos se realiza en sentido inverso a través de toda la red, lo que permite optimizar las conexiones entre neuronas.
- **Proceso iterativo de entrenamiento:** Durante el entrenamiento, los pasos de propagación hacia adelante, cálculo del error y retropropagación se repiten varias veces con diferentes muestras de datos, permitiendo que la red refine sus parámetros y aprenda patrones específicos.

2.9.3. Funciones de activación

Las funciones de activación son fundamentales para introducir no linealidad en el modelo. Ejemplos comunes incluyen la función *ReLU* (Rectified Linear Unit) y la sigmoide. Estas funciones determinan si una neurona se activa, lo que depende del valor ponderado de sus entradas.



2.10. Tipos de redes neuronales

Las redes neuronales pueden clasificarse en distintos tipos, cada uno diseñado para cumplir propósitos específicos. Aunque no se trata de una lista exhaustiva, a continuación se describen algunas de las variantes más comunes y sus casos de uso principales [2]:

2.10.1. Perceptrón

El perceptrón es la red neuronal más antigua, desarrollada por Frank Rosenblatt en 1958. Es el precursor de las redes neuronales modernas.

2.10.2. Perceptrón multicapa

También conocidas como *feedforward neural networks*, estas redes están formadas por una capa de entrada, una o más capas ocultas y una capa de salida. Aunque frecuentemente se les llama MLPs, técnicamente están compuestas por neuronas sigmoides y no por perceptrones simples, ya que los problemas del mundo real suelen ser no lineales. Estas redes se entrenan alimentándolas con datos y son la base de aplicaciones como visión por computadora, procesamiento de lenguaje natural y otros modelos avanzados.

2.10.3. Redes neuronales convolucionales

Las redes neuronales convolucionales (*Convolutional Neural Networks*, CNNs) son una variante de las redes de avance directo que se utilizan principalmente en tareas como el reconocimiento de imágenes, el análisis de patrones y la visión por computadora. Estas redes aplican principios del álgebra lineal, en particular la multiplicación de matrices, para identificar patrones dentro de las imágenes.

2.10.4. Redes neuronales recurrentes

Las redes neuronales recurrentes (*Recurrent Neural Networks*, RNNs) se caracterizan por tener lazos de retroalimentación en su estructura. Estas redes son especialmente útiles para trabajar con datos temporales, donde las predicciones dependen de la información previa. Por ejemplo, se aplican en la predicción de mercados financieros y en la proyección de ventas.

2.11. Limitaciones

Las RNA tienen la capacidad de capturar patrones complejos en grandes volúmenes de datos. Sin embargo, también presentan desafíos como [4]:

- **Necesidad de datos:** Requieren grandes cantidades de datos etiquetados para entrenarse adecuadamente.
- **Costo computacional:** Su entrenamiento puede ser intensivo en términos de tiempo y recursos computacionales.
- **Interpretabilidad:** Las predicciones de las RNA suelen ser difíciles de interpretar, lo que plantea retos en aplicaciones críticas.

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com

2.12. Reconocimiento facial

El reconocimiento facial es una técnica que permite a las computadoras predecir la identidad de una persona desde una imagen [1].

Tiene múltiples aplicaciones en la industria como el desarrollo de sistemas de asistencia, en la salud, sistemas de seguridad, etc [2].

2.12.1. Fundamentos del reconocimiento facial

A menudo, los términos verificación de rostros e identificación de rostros se usan de forma indistinta, sin embargo, aunque ambos comparten el mismo dominio del problema, lo abordan de forma distinta [3].

Para entender mejor la diferencia entre ambas tareas y su papel en el desarrollo de un sistema de reconocimiento facial, es que se presentan las técnicas más comunes que lo conforman [2].

Detección de rostros

En este paso se incluyen algunas estrategias de preprocesamiento que permiten obtener imágenes de buena calidad para los métodos de extracción de características. Aquí se localiza la región del rostro en una imagen o video y, dependiendo del contexto es posible aplicar técnicas de alineación o cortado de la imagen para aislar los elementos del fondo. Algoritmos como *Viola-Jones* y modelos basados en *Deep Learning*, como *Multi-task Cascaded Convolutional Networks* (MTCNN), son ampliamente utilizados para esta tarea [3].

Extracción de características

Una vez que se tiene localizado el rostro se procede a extraer las características de este. En sistemas tradicionales, solían utilizarse descriptores como Histogramas de Gradientes Orientados (*HOG*) o Escalas de Características Invariantes (*SIFT*). Actualmente, los modelos basados en las redes neuronales profundas, especialmente, las redes neuronales convencionales (CNN), como *FaceNet*, generan representaciones compactas conocidas como embeddings, que capturan la información más relevante del rostro [1].

Comparación y verificación

Los embeddings faciales obtenidos se comparan con una base de datos utilizando medidas de similitud como la distancia coseno o euclidiana. Este paso permite determinar si dos rostros corresponden a la misma persona (verificación) o identificar a una persona entre múltiples registros (identificación) [1].

2.12.2. Identificación de rostros

Como ya se mencionó, la identificación de rostros y la verificación de rostros son dos tareas distintas involucradas al momento de implementar un sistema de reconocimiento facial [1].

La identificación facial se refiere a la tarea de identificar la identidad de una persona dada una imagen.La imagen se ingresa por un extractor de características para obtener una representación f. Luego, esta representación, entra a una red de clasificación para asignar la identidad asociada a la imagen de entrada [2].

La identificación de rostros es una tarea de clasificación, por lo que los sistemas de reconocimiento facial basados en esta tarea son entrenados usando la función *softmax* y la pérdida de entropía cruzada [4].

2.12.3. Verificación de rostros

Contrario a la identificación de rostros, la verificación de rostros busca verificar que dos imágenes pertenecen a la misma entidad [1]. Por lo tanto, en un sistema de reconocimiento facial basado en la verificación de rostros recibe como entrada una imagen x y el identificador del rostro a ser identificado y la salida es una decisión binaria sobre si la imagen x pertenece a la persona con el identificador dado [2].

Este enfoque requiere de un proceso de extracción de características de los rostros de las personas que queremos que el sistema verifique y, por lo tanto, almacenar dichas representaciones en una base de datos [2].

2.12.4. Identificación mediante verificación

En los sistemas tradicionales de identificación, se entrena un clasificador con K clases para asignar a cada rostro de entrada una identidad específica de entre K personas en la base de datos. Sin embargo, este enfoque presenta limitaciones de escalabilidad, ya que al agregar una nueva persona, es necesario reentrenar completamente el sistema con $K + 1$ clases. Esto ocurre porque el clasificador inicial tiene K neuronas, pero para incluir una nueva identidad, se requiere un clasificador con $K + 1$ neuronas [2].

Para abordar este problema, un enfoque más eficiente y escalable es utilizar la comparación basada en similitud, propia de los algoritmos de verificación. En este caso, dado un rostro de entrada x, se ejecuta el algoritmo de verificación K veces, una para cada rostro almacenado en la base de datos. La identidad del rostro de entrada se asigna a la correspondiente al ID para el cual el algoritmo de verificación indica una coincidencia [2].

Por lo tanto, la ventaja de este tipo de enfoque híbrido radica en el hecho de que es posible agregar nuevas entidades al sistema sin la necesidad de re-entrenar los modelos de deep learning utilizados para la generación de los vectores de características de los rostros, más aún si se combina con algún tipo de mecanismo de almacenamiento de estos como las bases de datos vectoriales para acelerar el proceso de inferencia [1].

Av. Juan de Dios Bátiz esq. Miguel Othón de Mendizabal S/N Col. Lindavista, GAM, D. F. ☎ 57296000 Ext. 52045 ✉ ulises.velez@gmail.com